TARDEC

--- TECHNICAL REPORT ---

No. 21320

Baseline Field Testing of BB-2590 Lithium-Ion Batteries using an iRobot FasTac 510 Robot

By
Kevin Boice
Anthony Leo
Joseph Lee
John Paulson, Jr.
Matt Skalny
Ty Valascho

U.S. Army
Tank Automotive Research, Development, and Engineering Center
Detroit Arsenal
Warren, Michigan 48397-5000

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
17 SEP 2010	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Baseline Field Testing of BB-2590 Lithium-Ion Batteries using an iRobot FasTac 510 Robot	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Kevin Boice; Anthony Leo; Joseph Lee; John Paulson, Jr.; Matt Skalny; Ty Valascho	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397	**21340RC**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397	**TACOM/TARDEC**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
	21340RC

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

The views, opinion, and/or findings contained in this report are those of the authors and should not be construed as an Official Department of the Army position, policy, or decision, unless so designated by other documents., The original document contains color images.

14. ABSTRACT

Field testing of the 6.8 Ah BB-2590 Li-Ion Battery was performed to record and analyze power consumption and energy management characteristics as a baseline for future improvements. The testing was performed using an iRobot FasTac 510 robotic platform, using Mission 1 from the document Small Robot Mission Profiles V09."

15. SUBJECT TERMS

BB-2590, robot, battery, power, baseline

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**38**	

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
17-09-2010	Technical	04/2010 – 07/2010

4. TITLE AND SUBTITLE
Baseline Field Testing of BB-2590 Lithium-Ion Batteries using an iRobot FasTac 510 Robot

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kevin Boice, Anthony Leo, Joseph Lee, John Paulson, Jr., Matt Skalny, and Ty Valascho

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army, Tank Automotive Research Development and Engineering Command (TARDEC)
Warren, MI 48397

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army, Tank Automotive Research Development and Engineering Command (TARDEC)
Warren, MI 48397

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES
The views, opinion, and/or findings contained in this report are those of the authors and should not be construed as an Official Department of the Army position, policy, or decision, unless so designated by other documents.

14. ABSTRACT
Field testing of the 6.8 Ah BB-2590 Li-Ion Battery was performed to record and analyze power consumption and energy management characteristics as a baseline for future improvements. The testing was performed using an iRobot FasTac 510 robotic platform, using Mission 1 from the document "Small Robot Mission Profiles V09."

15. SUBJECT TERMS
BB-2590, robot, battery, power, baseline

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
UNCLAS DIST A					Ty Valascho
a. REPORT	b. ABSTRACT	c. THIS PAGE	A	33	19b. TELEPHONE NUMBER *(include area code)*
DIST A	DIST A	DIST A			(586) 282-0681

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Table of Contents

1. INTRODUCTION

Robots are frequently used by the Army for interrogation of Improvised Explosive Devices (IEDs) and occasionally for surveillance. These important roles help protect soldiers. Unfortunately, one significant limitation of robotics is battery life, especially in man-portable or "small" robots. If battery life can be extended, more possible missions could be realized. A baseline testing profile was created and measurements of voltage and current were taken while running a robot through this profile. This testing was performed on an iRobot FasTac 510 robot, using the latest approved BB-2590 Li-Ion batteries in April, May, and June 2010. Both products are currently used in theater by soldiers and provide a good baseline to measure future enhancements against. Another robotic vehicle, the QinetiQ TALON 4 robot is also used in theater and will be the subject of the next round of battery testing.

The BB 2590 Lithium Ion battery (National Stock Number 6140-01-490-4316), shown in Figure 1, is a rechargeable battery used throughout the Army to power many electronic devices, most commonly handheld radios. At 1.4 kg, its size and weight are such that it can be carried by the soldier in the field, including spares.

Figure 1 – Muddy BB-2590 Batteries after Field Testing

This battery is the preferred power source for robots used in the field, because of its durability, ubiquity, and performance characteristics. The US Army, through the Robotic Systems Joint Program Office (RS JPO) is actively working towards the goal of upgrading all man-portable military robots to use the BB-2590 battery. One such robot is the iRobot FasTac 510, photographed using two BB-2590 batteries as its power source in Figure 2.

Figure 2 – iRobot FasTac 510 Powered by two BB-2590 Li-Ion Batteries

As technologies are implemented in either batteries or robotic platforms to extend mission life, it is anticipated that the methodology and results contained in this report could be used as a baseline to measure the degree of improvement.

2. PURPOSE

The goal of this testing is to record and analyze the iRobot FasTac 510's energy-consumption and performance under a simulated "real world" mission. This simulated mission is designed to test and measure the robot's energy consumption on the various terrains experienced on deployment.

This report documents the findings from the baseline current and voltage measurements. It is intended to be used as a basis of comparison for improvements in the areas of power consumption and energy management for small robots.

The projected end-state is to have a robot with a longer lifespan that can meet the rigorous demands of the deployed soldier.

3. METHODOLOGY

The test runs were performed while logging data using custom python scripts that subscribe to data already available in the iRobot Aware 2.0 software of the FasTac 510. These custom scripts and modifications were made both on the robot and the Operator Control Unit (OCU). The data logging configuration is depicted in Figure 3, and described in more detail in Appendix C.

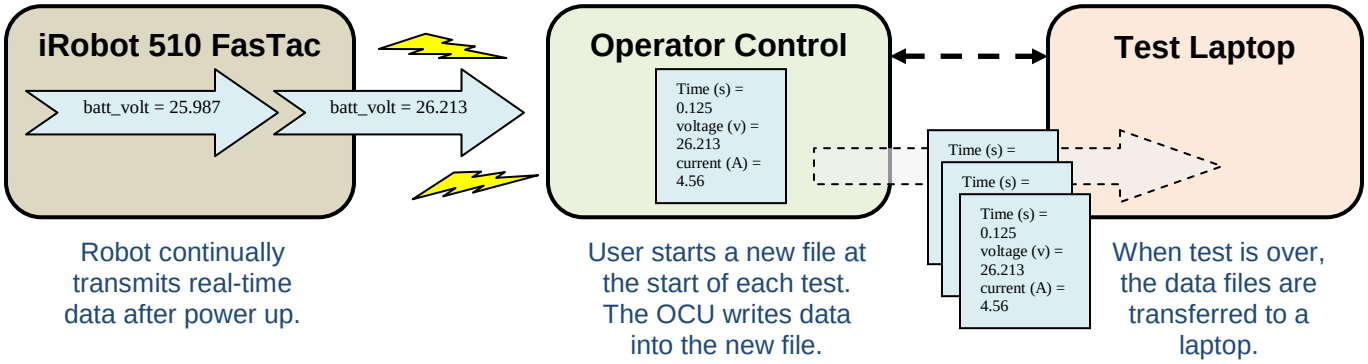

Figure 3 – Test Logging Configuration

Mission 1 from "Small Robot Mission Profiles V09" was used as the test plan for this work and is provided in Appendix A of this document. Appendix B provides additional testing details from this specific round of testing – procedures and best practices.

The test profile was run 3 times at each vehicle speed setting; creep, normal, and fast using two BB-2590 6.8 Ah Li-Ion batteries. Each test run is broken up into 6 subtests; Pea Gravel, Sand, Crushed Concrete, Hill, Obstacle Course, and Manipulator. It should be noted that the test site did not consist of the full length distances required for Mission 1, and the robot was turned around manually during the testing. For example, the Pea Gravel test calls for 100 m, but the test site only had 50 m of pea gravel.

An overhead view of the test site is provided in Figure 4.

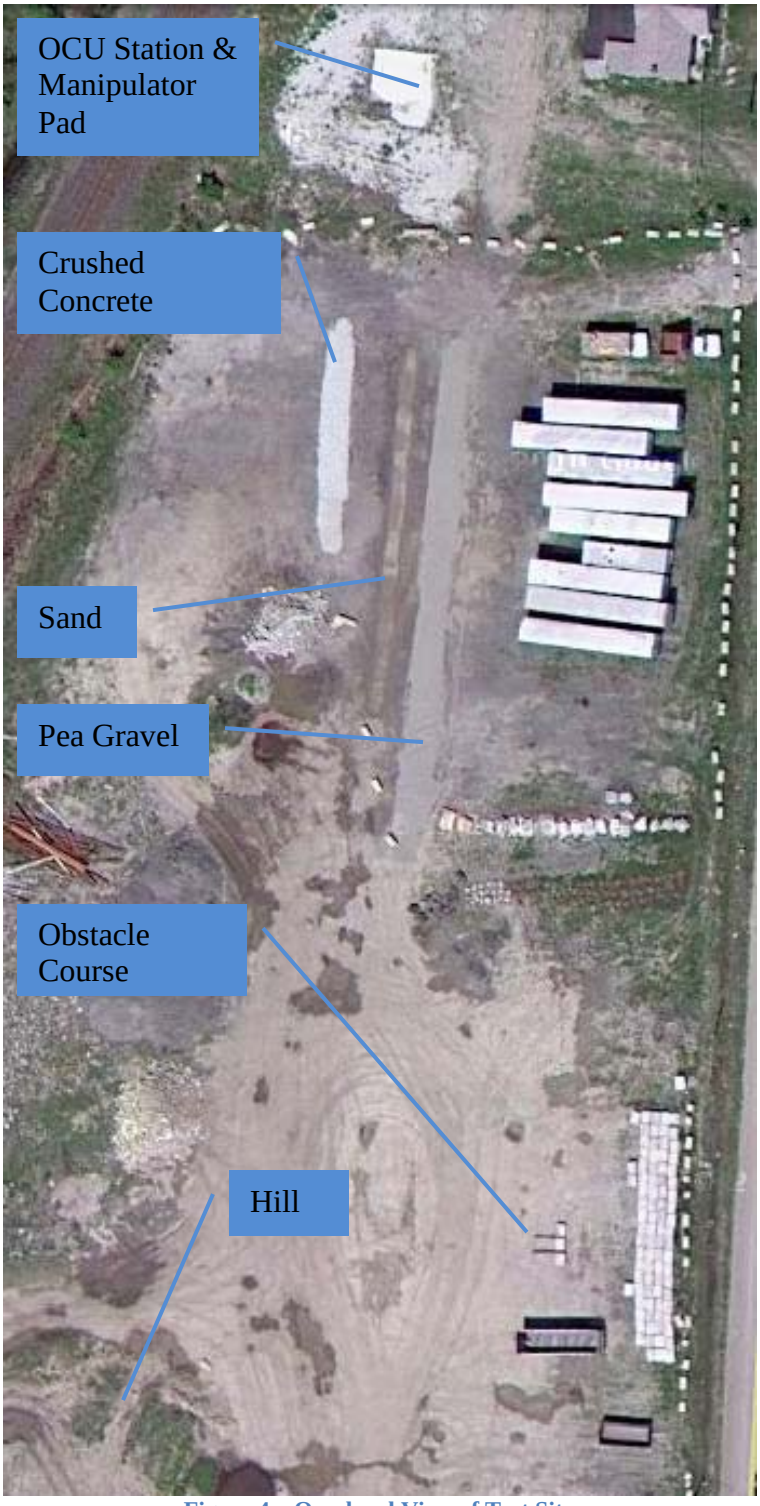

Figure 4 – Overhead View of Test Site

4. ANALYSIS OF THE DATA

The data logging system used for this testing creates comma separated values (csv) files with data values captured every 250 ms. These files do not have any header information, which means the file is all data values with no explanation as to what each value represents. A custom MS Excel Macro was created to process the data files and add the header information, the text of which is provided in Appendix E. After the raw csv files were processed, commercial software was used to analyze the data.

It must be noted that the data gathered during this testing came directly from the Aware 2.0 system. In some cases this data does not accurately represent the true hardware measurement. In particular, the Aware 2.0 values for battery voltage and power are made after the voltage regulator in the FasTac 510. This presents two issues with the measurements. The first is that, according to the manufacturer, the regulator is about 95% efficient resulting in some loss in the measurements. The second is the existence of the regulator itself, the function of which is to filter large transients and spikes from the rest of the system. The data therefore is regulated power measurements, which does not truly represent the raw values from the batteries themselves. The difference between the two measurements was examined by gathering data using both an external data logger connected to the battery output pins and the Aware 2.0 system over the same test run. From this comparison, it was determined that the Aware 2.0 values for the movement actuators - Left Track Current and Right Track Current – are closer to the measured values taken at the batteries. Therefore, all subtests except the Manipulator subtest used the Track Current values. The Manipulator subtest does not involve movement, so the post-processed Aware 2.0 values for battery voltage and power were used in this analysis.

Another issue that was discovered during testing was the idle current consumption. When the FasTac 510 is standing idle, it was found to consume an average current of 1.39 A to keep the processors, sensors, and system components powered. This means that during the subtests where the robot was manually turned around, it was consuming energy which should not be included in the analysis. A threshold was used to filter these time periods out of the data. It involved creation of a “Movement Bit”, which was used to represent whether the robot was being driven or was being manually turned around in the middle of a test. The Movement Bit is determined by comparing the xPosition and yPosition values every 250 ms with their previous values. If either the xPosition or yPosition changed by more than 0.02 from the previous measurement, the robot was considered “moving” and the Movement Bit was set to one. Otherwise, it was zero. In this way, only the time when the robot was moving was used for the final calculations. In addition, the periods of “non-movement” were used to calculate the idle current consumption value noted earlier.

A graph showing the Track Current values vs. the Battery Current values and the Movement Bit is provided in Figure 5 as an example. The value “Batt’y Current Measurements” is the sum of currents measured at the LeftFront battery and LeftRear battery. “Track Current Measurements” is the sum of currents measured at both track actuators. The Movement Bit is set to one when the platform is running the test and zero when idle. The 10 seconds in the middle of the test is where the platform was stopped after reaching the end of the sand terrain. This was only half the distance required for the test, though, so the platform was manually turned around at this point in time.

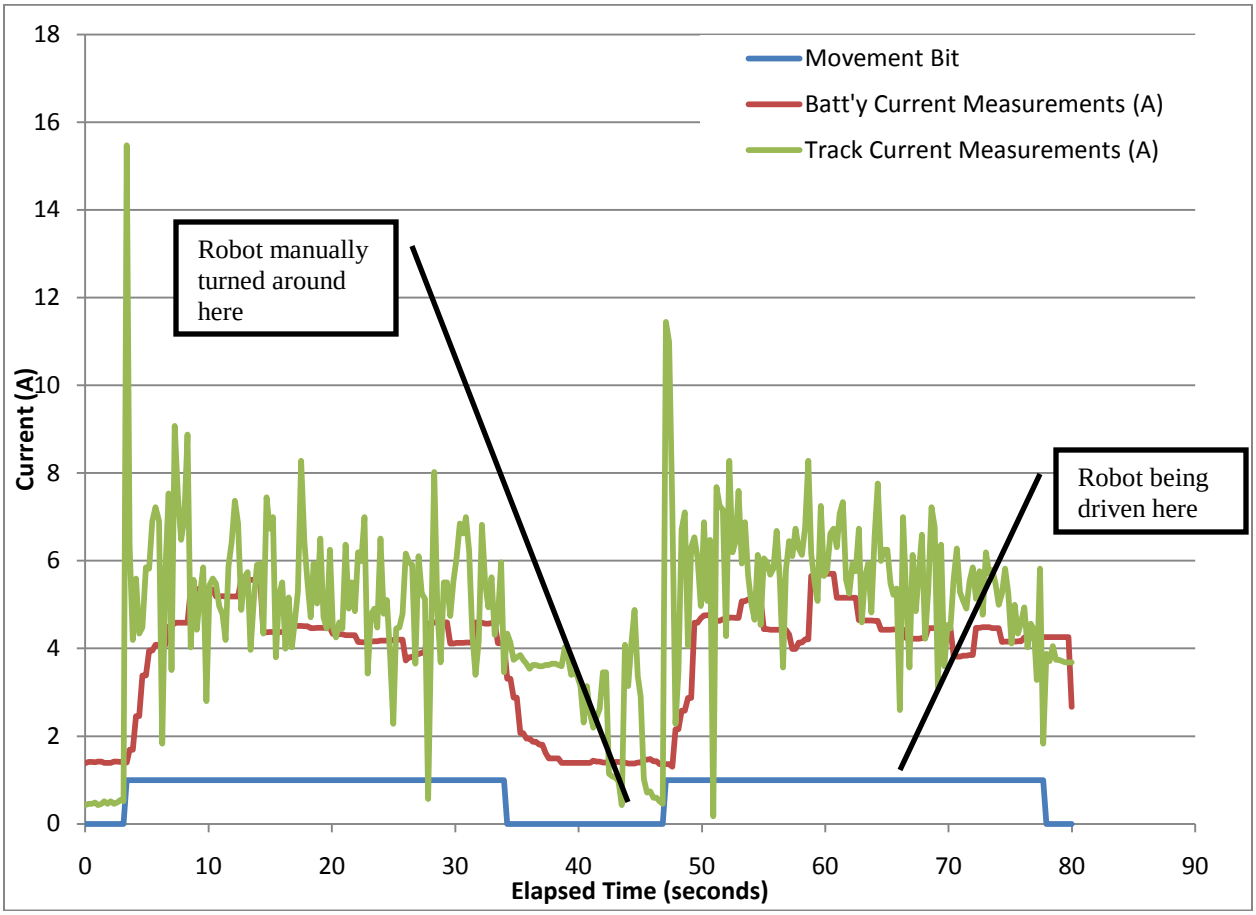

Figure 5 - Test 4.3 Sand Subtest Current Comparison

5. RESULTS

Table 1 presents the average current consumption during the battery field testing performed between April and June, 2010. All current data is the sum of Right Track Current and Left Track Current measurements, except the Manipulator subtest. The Manipulator subtest is the sum of the calculated LeftFront Battery Current and LeftRear Battery Current.

Table 1 – Summary of Average Current Consumption during Battery Field Test (Amps)

Subtest	Robot Speed Setting: Creep	Normal	Fast	Avg	Std Dev
Pea Gravel	4.74	5.34	6.84	5.64	1.08
Sand	5.04	5.42	6.70	5.72	0.87
Crushed Concrete	5.34	6.65	8.30	6.76	1.48
Hill	5.27	6.38	7.99	6.55	1.37
Obstacle Course	4.58	5.97	7.99	6.18	1.71
Manipulator	3.01	2.82	2.94	2.92	0.10
Avg	4.66	5.43	6.79	5.63	1.10

As expected, slower speed settings consume less average current.

The average time each test and subtest took to perform is provided in Table 2. These values are based upon the condition of the Movement Bit, except the Manipulator subtest. That is, the Time of Test shown in this table is only the time that the robot was moving during the test – it does not include idle time. In the case of the Manipulator test, it is a timed test of exactly 5 minutes and the Movement Bit never goes to one as the platform is stationary during the entire test.

Table 2 – Average Time of Test during Battery Field Test (seconds)

Subtest	Robot Speed Setting: Creep	Normal	Fast	Avg
Pea Gravel	571.5	102.7	40.0	238.1
Sand	341.9	60.1	25.3	142.4
Crushed Concrete	112.8	20.1	9.1	47.3
Hill	549.9	103.0	51.3	234.7
Obstacle Course	112.5	21.9	13.9	49.4
Manipulator	305.8	306.3	305.2	305.7
Total	1994.4	614.0	444.7	169.6

In Figure 6, an “idealized” plot of regulated battery voltage versus time for a complete test run is presented. This plot is a combination of the subtest recordings that are closest to the average current consumption of each subtest. It gives a general representation of the supply voltage profile that any equipment attached to a FasTac 510 can expect.

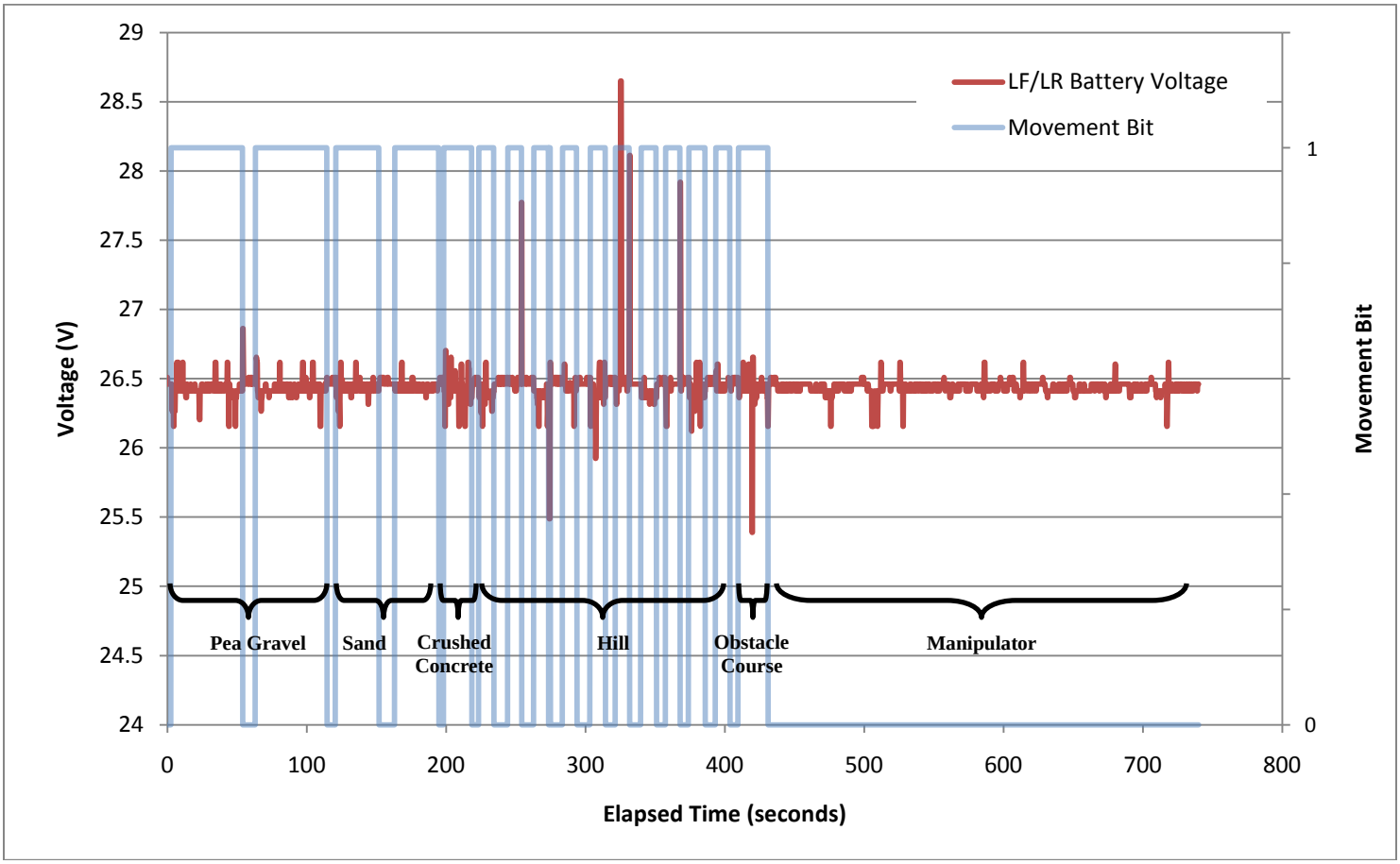

Figure 6 – Idealized Combination of Average Voltage Plots

Selected graphs of the current consumption over time for each subtest are presented in Figure 7 through Figure 12. These current traces are from the same specific subtests as the combined subtest plots of voltage in Figure 6. That is, the Pea Gravel subtest from test run 4.3 was used for both the Pea Gravel voltage plot in Figure 6 and the current consumption graph in Figure 7.

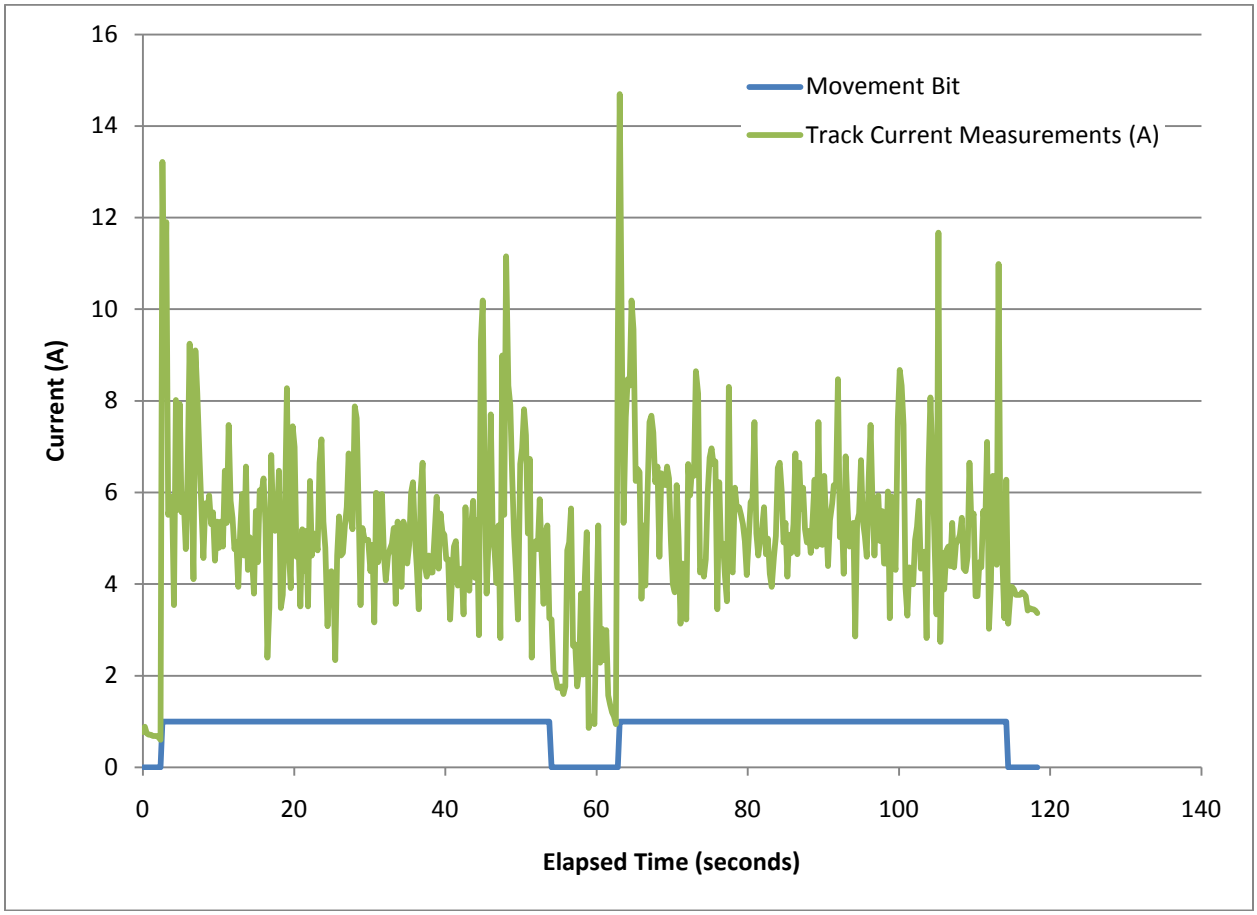

Figure 7 - Test 4.3 Pea Gravel Subtest Current Consumption

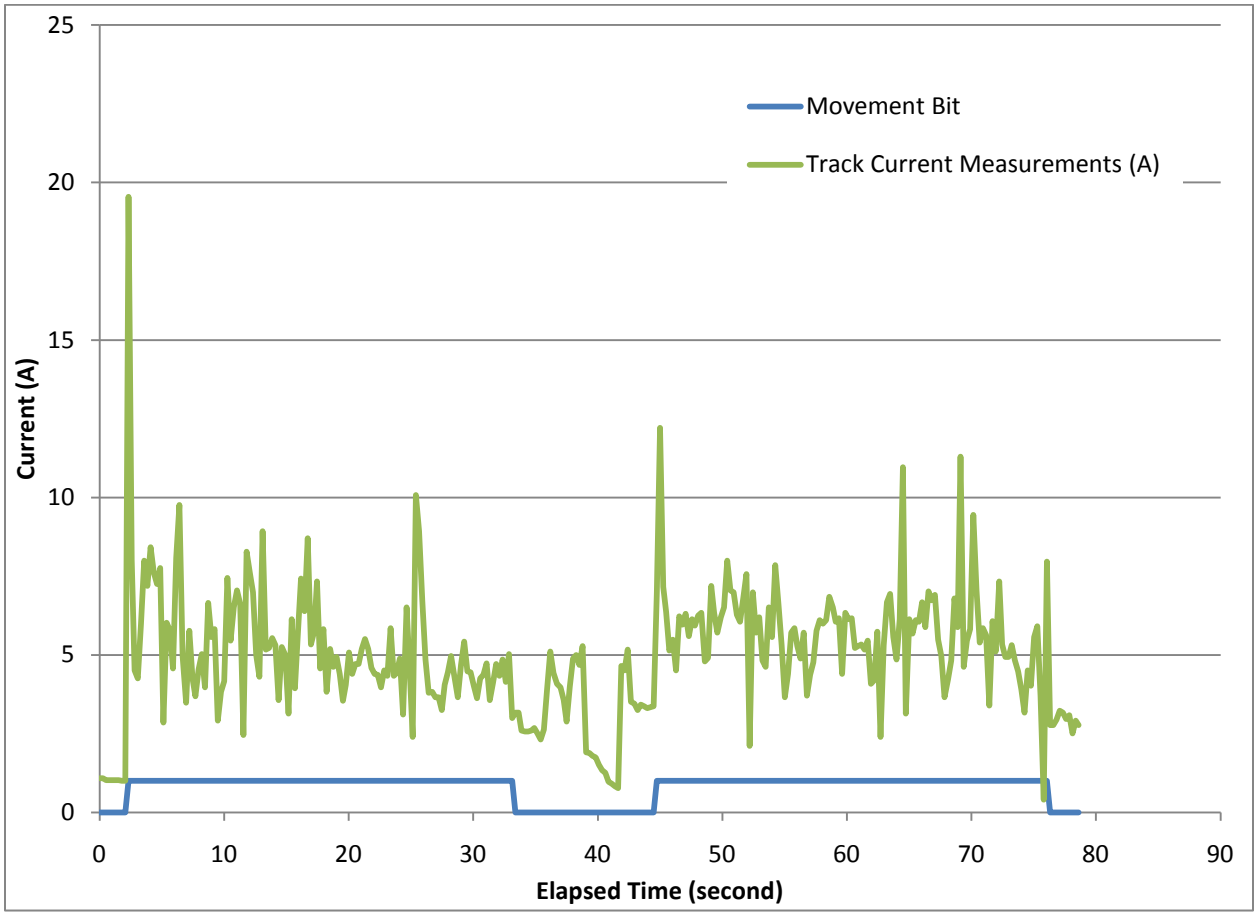

Figure 8 - Test 4.2 Sand Subtest Current Consumption

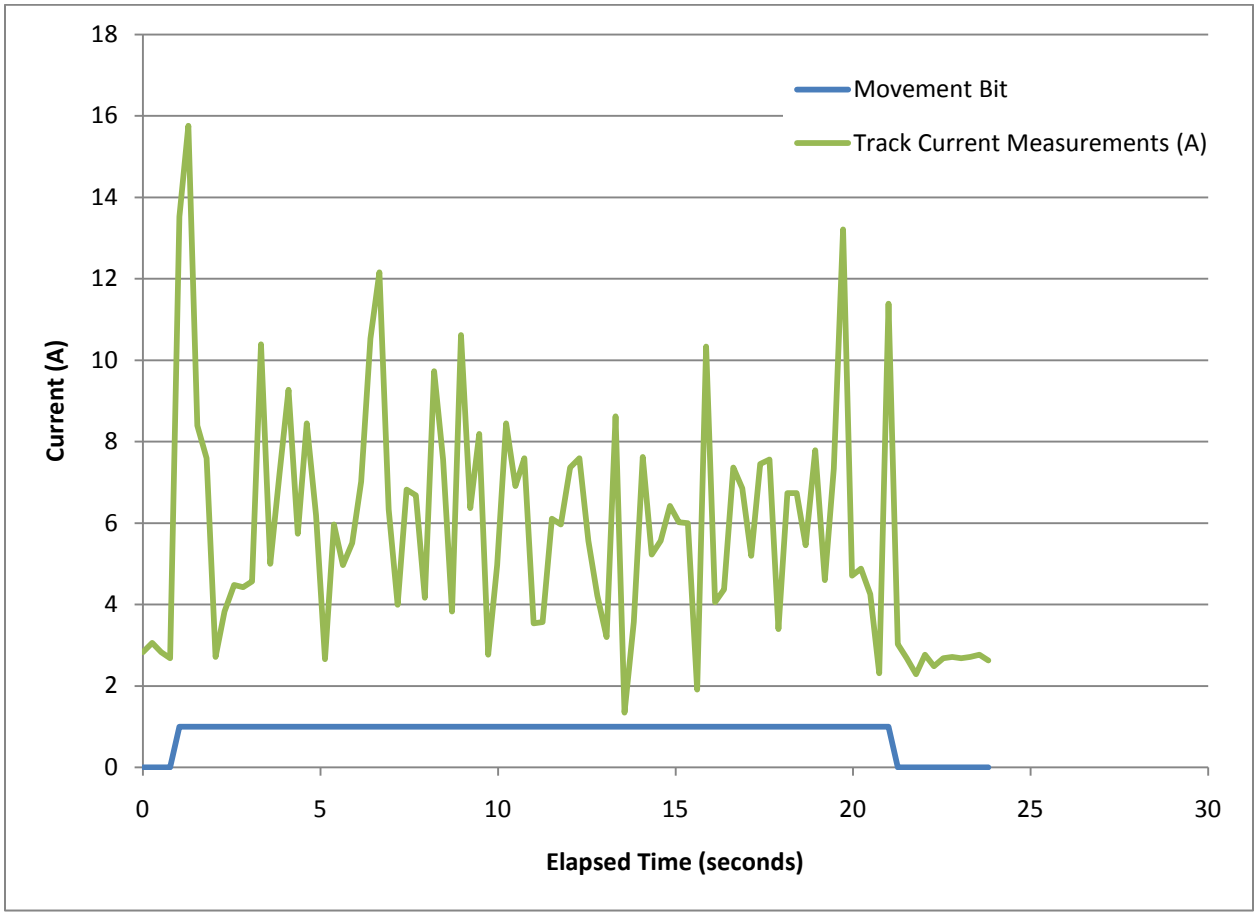

Figure 9 - Test 4.3 Crushed Concrete Subtest Current Consumption

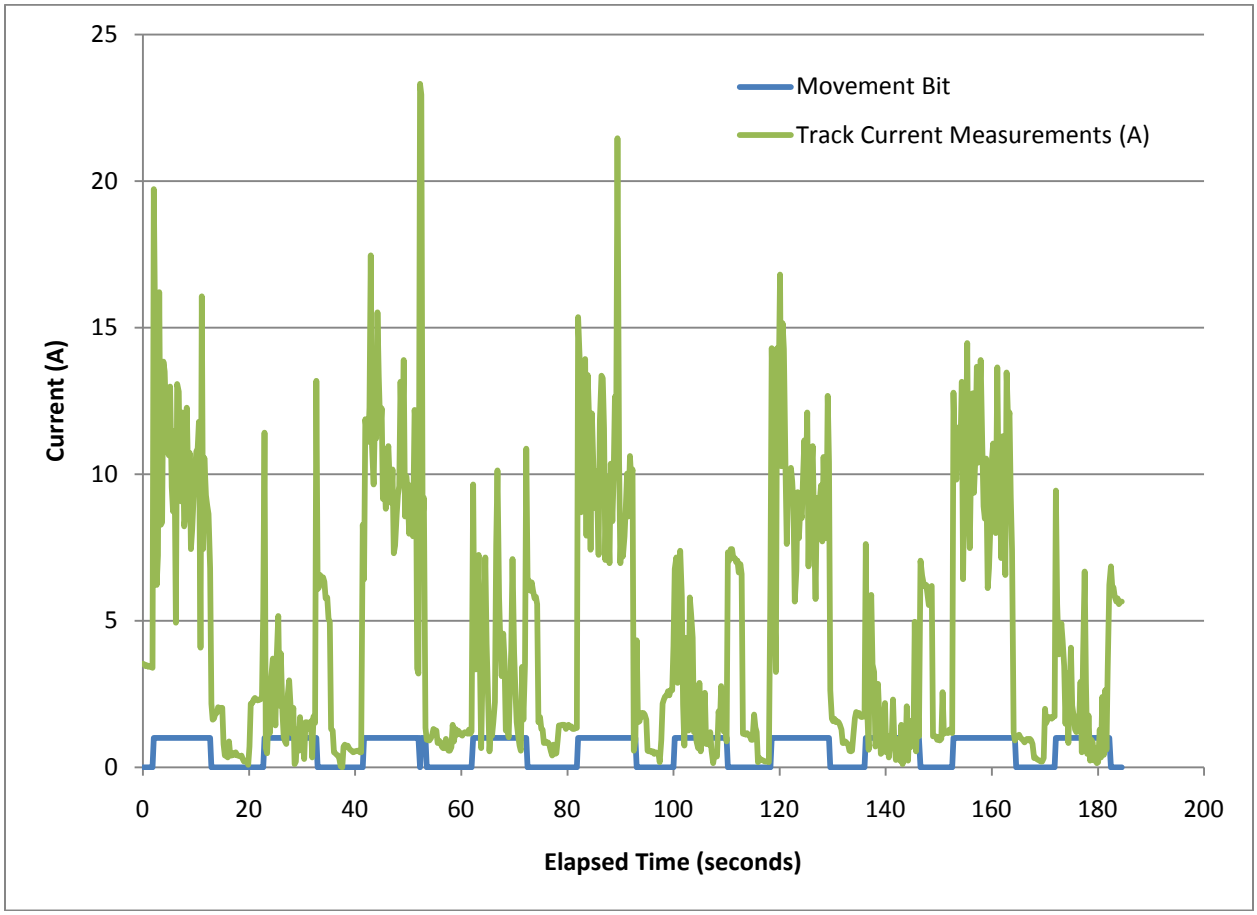

Figure 10 - Test 4.3 Hill Subtest Current Consumption

In Figure 10, there are 10 distinct movement events in the subtest that correspond to the five up and down segments of the Hill subtest. Current consumption is much higher while climbing the hill, so it can be deduced from the graph that the robot is driven up the hill first, then down, with this cycle repeated until the correct distance was driven.

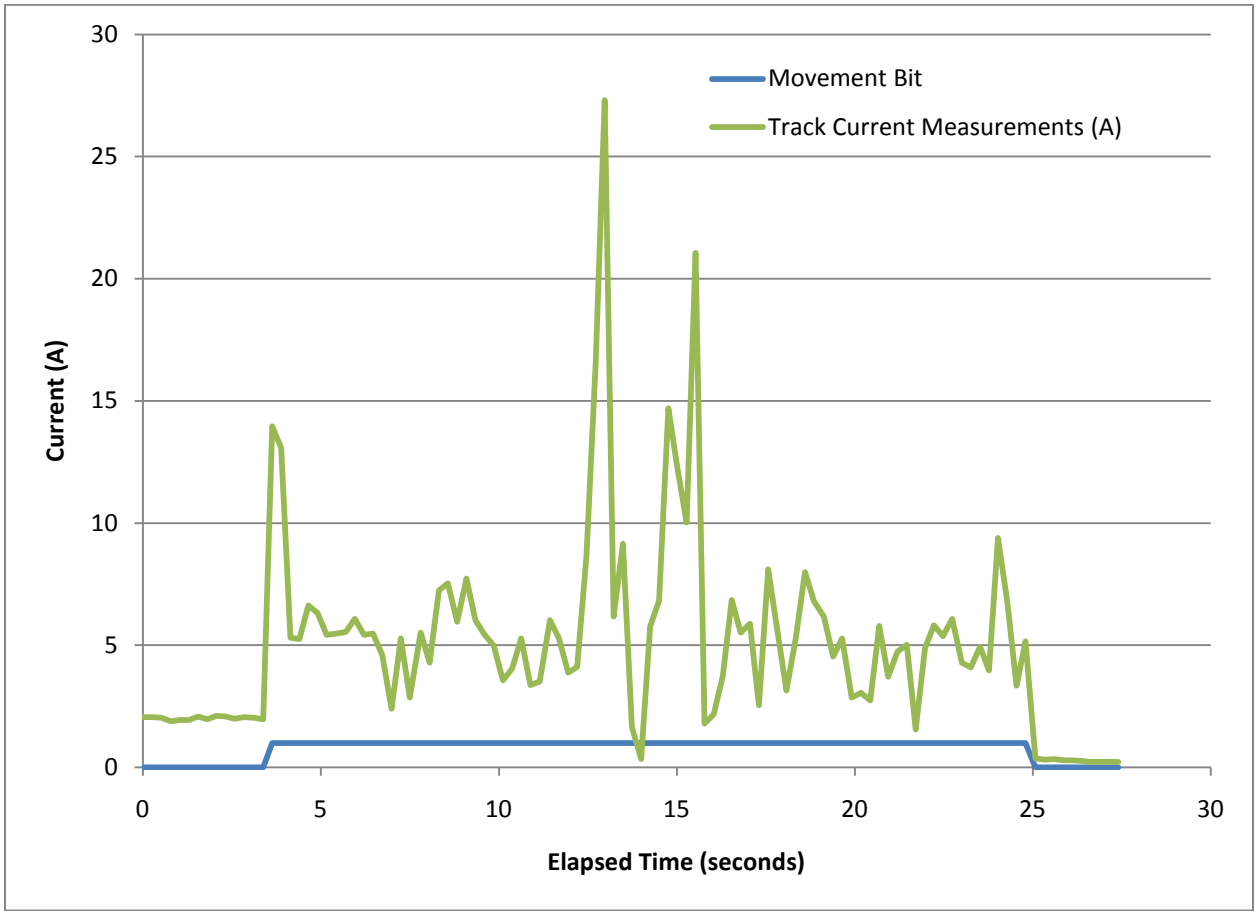

Figure 11 - Test 4.2 Obstacle Course Subtest Current Consumption

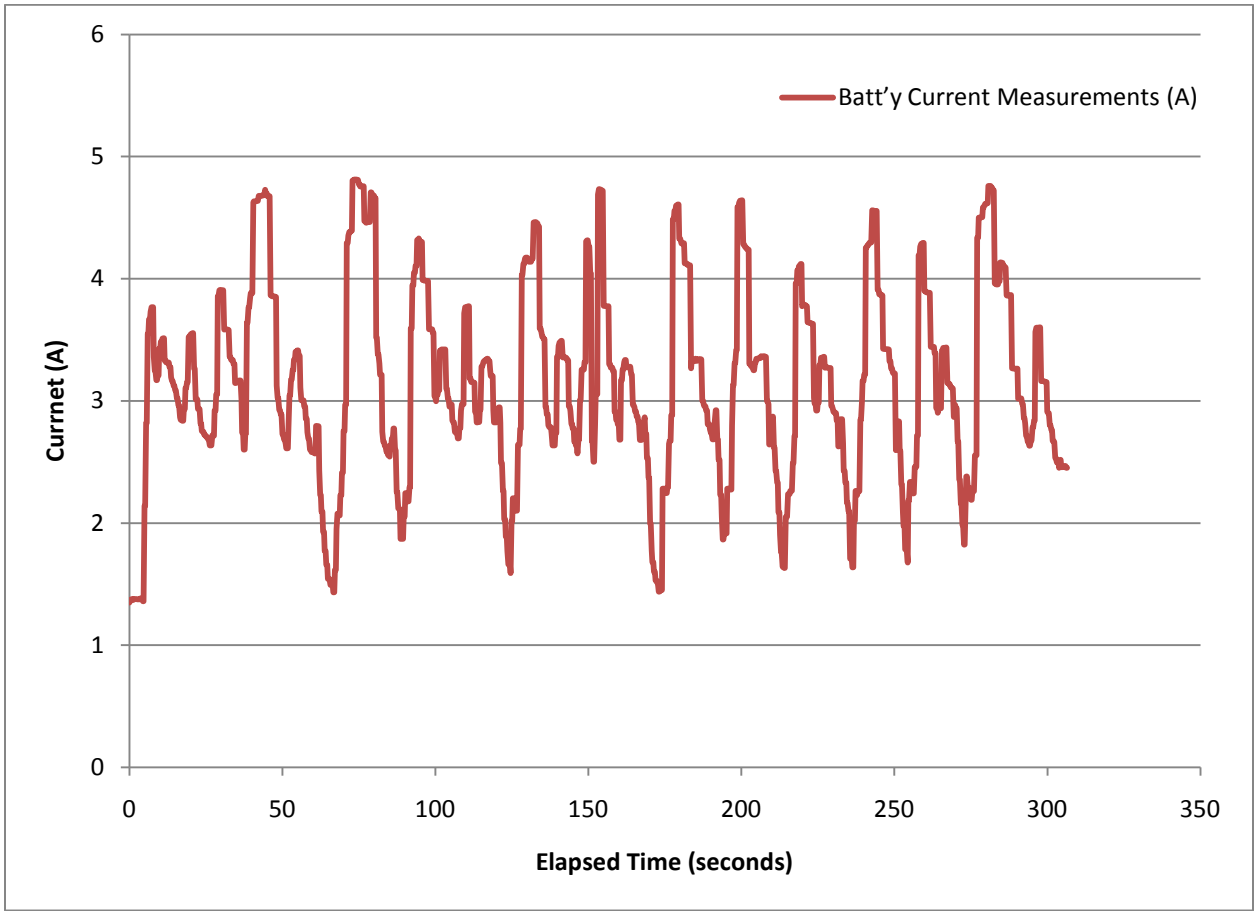

Figure 12 – Test 4.3 Manipulator Subtest Current Consumption

It should be noted that in Figure 12, the LeftFront and LeftRear Battery Currents are plotted, not the track currents. As explained previously, this is because the Manipulator Subtest does not involve movement, so the track current measurements were not used.

Disclaimer for Publications

****Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.****

APPENDIX A. MISSION 1 FROM "SMALL ROBOT MISSION PROFILES V09"

Mission 1

Background

This mission profile combines the basic tasks of the following in-theater missions:

- Short-range Surveillance
- Short-range Reconnaissance
- Improvised Explosive Device (IED) investigation
- Checkpoint inspection
- Route clearance
- Engineering detonation in place

These are all currently Line of Sight (LOS) missions or teleoperated applications.

Profile

All traveling is done at an average speed of 4 kph, while transmitting full color video back to the OCU. If run continuously, this entire test should be completed within 40 minutes. It is acceptable to run the test non-continuously (piecemeal), but each step should be kept intact and the power source(s) must not be replenished between steps.

OCU shall be kept at standoff distance of approximately 100 m from the robot during the test.

Ambient temperature of 25° C, +/- 5° C, clear conditions, less than 10 kph wind.

1. Robot picks up a 2.0 kg weight and carries it.
2. Robot travels 100 m over a pea gravel road gravel surface.
3. Robot travels 60 m on loose, sandy surface.
4. Robot travels 20 m over rocks or crushed concrete of an average size of approximately 75 mm diameter.
5. Robot travels 10 m uphill on grass or dirt at 30% slope. The total height will be approximately 4.5 m.
6. Robot travels 10 m downhill (negative obstacle) on grass at 30% slope. The total depth will be approximately 4.5 m.
7. Repeat steps 5 and 6 four more times for a total of five times (100 m of total distance traveled).
8. Robot drops 2.0 kg weight.
9. Robot travels 20 m over an asphalt or concrete surface and climbs over two obstacles 0.1 m in height and spaced 9 m apart and approximately 0.2 m in width, as seen in
10. Table 3.

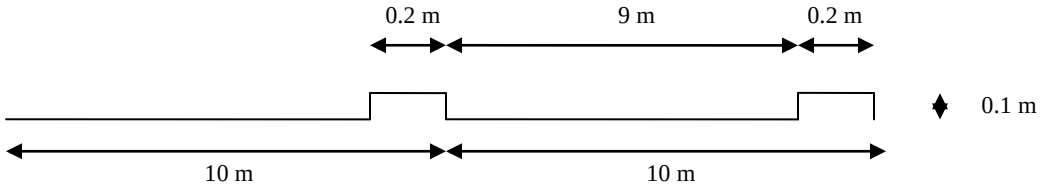

Table 3: Mission 1 Step 7 Obstacle Diagram

11. Robot manipulates arm for 5 minutes by continuously executing set poses, if available, or some repeatable motion sequences otherwise.

APPENDIX B. DETAILED TESTING PROCEDURE FOR IROBOT FASTAC 510

Equipment

- FasTac 510 robot
- OCU
- Two BB-2590 6.8 Ah batteries, Nat'l Stock Number 6140-01-490-4316
- Test Laptop with interface to the OCU, i.e. ethernet and PuTTY
- 2.0 kg test weight
- Multiple hard copies of Test Sheet
- Generator (for OCU and laptop)
- Surveyor's Wheel
- Handheld 2-Way Radios
- Stopwatch

Procedures

Once the test site is set up, place a station with the OCU and Test Laptop near the test tracks. Before running any tests, ensure that RF communication is maintained between the robot and the OCU at every test location. This may require that every test area have Line of Sight to the OCU station.

There are three distinct job responsibilities for this test. One person is designated the robot Driver, and is responsible for starting and stopping the test log file and also driving the robot. A second person, the Observer, records general observations during testing, maintains the Test Sheet, and verifies that log files have been stopped and started correctly on the OCU from the Test Laptop. In addition, the Observer operates a 2-way handheld radio to communicate with the third person, the Spotter. The Spotter will carry a handheld 2-way radio and walk with the robot during the test.

Before putting the batteries in the robot, the Observer fill out the test sheet, supplied in

Appendix D. The iRobot OCU serial number is located on a sticker on the bottom left hand corner of the keyboard. The battery model and serial number are located in yellow on the side of the battery. The five digit serial number of the robot is located next to the power button, in the rear by the handle.

Connect the robot to the OCU, after turning both on. Push and hold the robot power button until the green light turns on. The OCU starts up like a normal laptop. Once at the main menu, enter the FasTac 510 serial number to connect the OCU by wireless. If there is any trouble powering up or connecting the OCU or robot, refer to the iRobot FasTac 510 Operators Manual.

Verify that the Test Laptop and OCU are connected and that PuTTY is running on the Test Laptop as an interface to the Linux file system.

For portions of the test, the robot carries a 2.0 kg (4.4 lb) test weight. It was found during testing that a sturdy plastic jug with a handle works very well as the test weight. This jug is filled with gravel or rocks to the desired weight. Small holes frequently appear in the jug during testing, so water and sand do not work as well as larger materials. Examples of test weights are provided in Figure 13.

Figure 13 – Test Weights Used for Battery Field Testing

Have the robot carry the weight with its gripper. Position the driving cameras so that the weight does not obscure the driver's vision. It is also preferred to position the weight and arm to change the robot's center of gravity as little as possible. The preferred driving position for the FasTac 510 is the "Pick Up or Place Objects" pose.

For each subtest, a new test file should be started and the name of this file recorded on the Test Sheet. It is crucial that the file names are correctly recorded on the Test Sheet, because this is the only place where the correlation between log file and subtest is made. If an error occurs during any part of the subtest, simply stop that log file, record that the file is unusable on the test sheet, and restart that subtest with a new log. It is useful to have the Spotter tell the driver when distance markers have been reached, as this can be difficult for the Driver to discern.

The manipulator subtest can be performed on any flat surface. A convenient location is close to the OCU station. Be sure to record any position failures or problems with the actuators during the test.
When one complete test run is finished – all six subtests - the log files should be transferred to the Test Laptop for distribution and analysis.

APPENDIX C. PYTHON SCRIPTS AND SETUP PROCEDURE FOR LOGGING DATA USING IROBOT AWARE 2.0

Equipment

- FasTac 510 robot with Aware 2.0 and Integrator Edition version 3.4
- FasTac OCU or Other Logging Computer
- Established connection between FasTac 510 robot and OCU/Other Logging Computer

Procedures

Overview

There are two python scripts needed for logging battery data from the PackBot FasTac 510 (robot) to a remote computer. The remote computer can be any computer that is connected via the wireless or wired link to the robot, but for the purposes of this test, an Aware 2.0 iRobot OCU was used. A script called fullbatterychar.py ran on the robot collecting battery, orientation, commanded motion, and motor information. A script called ocubatterylogger.py ran on the OCU and logged information that was sent by the fullbatterychar.py script. The robot is running a Linux variant.

Setting Up the PackBot FasTac 510

There are two steps required to properly set the PackBot FasTac 510 (robot) for logging information required for battery testing. First, the fullbatterychar.py python script must be installed on the robot. This is done by placing the script (see end of this appendix for full script) in a location on the robot that it will be run from and changing the TARGET_IP value within the fullbatterychar.py script. For this test, the location /opt/tardec/scripts was used, but any location with proper permissions is acceptable. The TARGET_IP variable must be set to the IP address of the remote machine that the ocubatterylogger.py script is running on. **Make sure that the Integrator Edition version on the robot is 3.4 – the script may not function properly on other versions due to discrepancies in publication names.** After the script is installed in the desired location, it must be started by running the command "python /opt/tardec/scripts/fullbatterychar.py &" from the shell prompt, substituting "/opt/tardec/scripts/" with the location the script is in if necessary. Alternatively, the script can be started automatically by adding the line "python /opt/tardec/scripts/fullbatterychar.py" to the end of the "startAll()" function in the /etc/rc.d/init.d/aware2 file. This will start the script after all Aware2 processes have been started.

Setting Up the Remote (Logging) Computer

The logging computer can be any computer with python installed. For this test, the logger script was placed on the iRobot Aware 2.0 OCU computer, but in general the script can be run on any computer that has a connection to the robot computer and has an IP address equal to that specified in the TARGET_IP variable in the fullbatterychar.py script running on the robot. If running on a computer other than the iRobot Aware 2.0 OCU, simply place the ocubatterylogger.py in the desired location and run it through a python interpreter – the script will listen for UDP packets coming from the fullbatterychar.py script on port 20000. The logger can be run through the python interpreter as follows (this can be made into a separate script):

```
>>> import ocubatterylogger
>>> logger = ocubatterylogger.OcuBatteryLogger()
>>> logger.start()
… go until logging done …
>>> logger.stop() or ctrl-c process
```

The packets are in a comma delineated string format, and are logged directly to a file called battery-[timestamp].csv in the directory "/home/tardec/". The file location and naming scheme can be changed in the

ocubatterylogger.py script by modifying the line *filename = (‘/home/tardec/battery-‘ + time.asctime() + ‘.csv’)* to reflect the desired location and naming convenction for storing the csv files.

The ocubatterylogger.py logger can also be run by clicking on a custom created button on the iRobot Aware 2.0 OCU screen. To enable this capability, three steps need to be taken on the iRobot Aware 2.0 OCU:

1. Place the ocubatterylogger.py script in the /opt/irobot/lib/python2.5/site-packages/cpOcuApp/fastac folder.
2. Modify the /opt/irobot/lib/python2.5/site-packages/cpOcuApp/ocuApp/MainMenuBase.py file to add a new button to turn the battery logger on and off.
3. Modify the /opt/irobot/lib/python2.5/site-packages/cpOcuApp/FasTacSessionScreen.py file to create and start/stop the battery logger thread when a button is pressed.

In this way, the same OCU that is used to control the robot can also be used to do the logging without having to manually start and stop the ocubatterylogger.py script.

fullbatterychar.py script (10 May 2010 version)

```
#!/opt/tardec/bin/python
import aware
import aware.rf
from time import sleep
import time
import socket

# Author: Matthew Skalny
# Description: This is a script designed to get battery, temperature,
# and steering commands off the PackBot (aware 2 fastac).  It uses
# pub/sub on the robot, and then utilizes UDP packets sent back to
# OCU to communicate the data to the logger.

def addData(currentData, value, isvalid):
    if isvalid == True:
        return currentData + str(value) + ","
    else:
        return currentData + 'Invalid' + ","

def getBatterySubData(sub):
    data = ''
    try:
        batteryValue = sub.getAny(True).resolve()
        # Extract the requried information from the batteries
        leftFrontBattery = batteryValue.find("LeftFront")
        leftRearBattery = batteryValue.find("LeftRear")

        # Get left front battery information
        leftFrontVolts = leftFrontBattery.volts.value.value
        data = addData(data, leftFrontVolts, leftFrontBattery.volts.value.grade.isValid()
== 1)
        leftFrontEnergy = leftFrontBattery.energy.value.value
        data = addData(data, leftFrontEnergy,
leftFrontBattery.energy.value.grade.isValid() == 1)
        leftFrontPower = leftFrontBattery.power.value.value
        data = addData(data, leftFrontPower, leftFrontBattery.power.value.grade.isValid()
== 1)
        leftFrontCellTemp = leftFrontBattery.cellTemperature.value
        data = addData(data, leftFrontCellTemp,
leftFrontBattery.cellTemperature.grade.isValid() == 1)
```

```
        # Get left rear battery information
        leftRearVolts = leftRearBattery.volts.value.value
        data = addData(data, leftRearVolts, leftRearBattery.volts.value.grade.isValid()
== 1)
        leftRearEnergy = leftRearBattery.energy.value.value
        data = addData(data, leftRearEnergy, leftRearBattery.energy.value.grade.isValid()
== 1)
        leftRearPower = leftRearBattery.power.value.value
        data = addData(data, leftRearPower, leftRearBattery.power.value.grade.isValid()
== 1)
        leftRearCellTemp = leftRearBattery.cellTemperature.value
        data = addData(data, leftRearCellTemp,
leftRearBattery.cellTemperature.grade.isValid() == 1)
        return data
    except Exception:
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        return data

def getOrientationSubData(sub):
    data = ''
    try:
        # Get the roll/pitch/yaw information
        orientationValue = sub.getAny(True).resolve()
        valid = sub.isValidValue()
        hpr = orientationValue.getHPR()
        heading = hpr.heading
        data = addData(data, heading, valid)
        pitch = hpr.pitch
        data = addData(data, pitch, valid)
        roll = hpr.roll
        data = addData(data, roll, valid)
        return data
    except Exception:
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        return data

def getCommandSubData(sub):
    data = ''
    try:
        # Get the translate and rotate information (translate from x, rotate theta)
        commandValue = sub.getAny(True).resolve()
        valid = sub.isValidValue()
        translate = commandValue.find("X").velocity
        data = addData(data, translate, valid)
        xposition = commandValue.find("X").position
        data = addData(data, xposition, valid)
        yposition = commandValue.find("Y").position
        data = addData(data, yposition, valid)
        rotate = commandValue.find("Theta").velocity
        data = addData(data, rotate, valid)
        return data
```

```
    except Exception:
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        data = addData(data, "N/A", True)
        return data

def getMotorVoltsSubData(sub):
    try:
        # Get the motor volts
        motorVoltsValue = sub.getAny(True).resolve()
        cam = motorVoltsValue.find("CAM")
        flipper = motorVoltsValue.find("Flipper")
        sam = motorVoltsValue.find("SAM")
        tracks = motorVoltsValue.find("Tracks")
        tracksVolts = tracks.value.value
        flipperVolts = flipper.value.value
        samVolts = sam.value.value
        camVolts = cam.value.value
        data = str(camVolts) + "," + str(flipperVolts) + "," + str(samVolts)
        data += "," + str(tracksVolts)
        return data
    except Exception:
        data = ("N/A" + "," + "N/A" + "," + "N/A" + "," + "N/A")
        return data

def getMotorCurrentsSubData(sub):
    try:
        motorCurrentsValue = sub.getAny(True).resolve()
        trackLeft = motorCurrentsValue.find("Track Left")
        trackRight = motorCurrentsValue.find("Track Right")
        trackLeftCurrent = trackLeft.value.value
        trackRightCurrent = trackRight.value.value
        data = str(trackLeftCurrent) + "," + str(trackRightCurrent)
        return data
    except Exception:
        data = ("N/A" + "," + "N/A")
        return data

def createSub(subName, subType):
    sub = aware.Subscription.newInstance(subName, subType)
    return sub

#================================================================================
# MAIN SCRIPT
#================================================================================

# create the one and only aware2 module
batteryMod = aware.Module.instance()
batteryMod.init(["-name", "BatteryCharacterization"])

# Constants for subscription types and names.
BATTERY_SUB_TYPE = "aware::rf::MultiBatteryState"
ORIENTATION_SUB_TYPE = "aware::rf::CompassData"
COMMAND_SUB_TYPE = "aware::rf::MultiAxisState"
MOTOR_VOLTS_SUB_TYPE = "aware::rf::MultiFloatState"
MOTOR_CURRENTS_SUB_TYPE = "aware::rf::MultiFloatState"

BATTERY_SUB_ALIAS = "/chassis/batteries"
ORIENTATION_SUB_ALIAS = "/chassis/orientation"
COMMAND_SUB_ALIAS = "/chassis/odometry"
```

```
MOTOR_VOLTS_SUB_ALIAS = "/combinedMotion/motorVolts"
MOTOR_CURRENTS_SUB_ALIAS = "/combinedMotion/motorCurrents"

# Other constants
SLEEP_TIME_SEC = 0.25

# Create local subscriptions
batterySub = createSub("batterySub", BATTERY_SUB_TYPE)
orientationSub = createSub("orientationSub", ORIENTATION_SUB_TYPE)
commandSub = createSub("commandSub", COMMAND_SUB_TYPE)
motorVoltsSub = createSub("motorVoltsSub", MOTOR_VOLTS_SUB_TYPE)
motorCurrentsSub = createSub("motorCurrentsSub", MOTOR_CURRENTS_SUB_TYPE)

# Insert the subscription into the module
batteryMod.insert(batterySub)
batteryMod.insert(orientationSub)
batteryMod.insert(commandSub)
batteryMod.insert(motorVoltsSub)
batteryMod.insert(motorCurrentsSub)

# Set Subscriptions usable
batterySub.setUsable()
orientationSub.setUsable()
commandSub.setUsable()
motorVoltsSub.setUsable()
motorCurrentsSub.setUsable()

# Connect the subscriptions
batterySub.connect(BATTERY_SUB_ALIAS)
orientationSub.connect(ORIENTATION_SUB_ALIAS)
commandSub.connect(COMMAND_SUB_ALIAS)
motorVoltsSub.connect(MOTOR_VOLTS_SUB_ALIAS)
motorCurrentsSub.connect(MOTOR_CURRENTS_SUB_ALIAS)

# Create socket for sending, including defines for send address and port
TARGET_IP = "172.17.168.106"
#TARGET_IP = "172.16.85.200"
TARGET_PORT = 20000
sendSocket = socket.socket( socket.AF_INET, socket.SOCK_DGRAM)

# Enter the to record data and send it over the network (UDP)
recordData = True
iterations = 720000
while recordData == True:
    # Print off the data we are packing.
    # lf (volts energy power temp), lr(volts energy power temp)
    # heading pitch roll translate rotate1
    dataTime = time.time()
    data = str(dataTime) + "," + getBatterySubData(batterySub)
    data += getOrientationSubData(orientationSub)
    data += getCommandSubData(commandSub)
    data += getMotorVoltsSubData(motorVoltsSub) + ","
    data += getMotorCurrentsSubData(motorCurrentsSub)

    # Pack the data for transmission
    # First run - just use an already formatted csv string and send it
    # Send the data over UDP
    sendSocket.sendto(data, (TARGET_IP, TARGET_PORT))

    # Sleep .25 seconds - send 4 times per second
    time.sleep(SLEEP_TIME_SEC)
```

```
    iterations-=1
    if iterations == 0:
        recordData = False

sendSocket.close()
```

ocubatterylogger.py script (10 May 2010 version)

```
#!/opt/tardec/bin/python
from __future__ import with_statement
from time import sleep
import time
import socket
import threading

# Author: Matthew Skalny
# Description: This is a script designed to receive a csv line over UDP from
# a battery logger residing on the PackBot FasTac.  This logger simply
# listens for incoming messages on the specified port until it receives one,
# which it then writes directly to a file specified as a new line.

class OcuBatteryLogger (threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.LISTEN_PORT = 20000
        self.LISTEN_IP = ""

        self.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.sock.bind( (self.LISTEN_IP, self.LISTEN_PORT) )
        self.sock.setblocking(False)

        self.continueLogging = True

        # Lock for starting/stopping thread
        self.lock = threading.Lock()

    # This function is a function that will create a new file for storing
    # csv formatted data.  The file is placed in the specified directory
    # and is guaranteed a unique name by using the current time from the
    # epoch within the file name.  This function should be called every time
    # it is desired to start a new logging session.
    def recordData(self):
        timeForFile = time.time()
        filename = ('/home/tardec/battery-' + time.asctime() + '.csv')

        with open(filename, 'w') as f:
            while True:
                self.lock.acquire()
                if self.continueLogging == False:
                    self.lock.release()
                    return
                self.lock.release()
                try:
                    data, addr = self.sock.recvfrom(4096)
                    # write the data to the file
                    f.write(data + '\n')
                    time.sleep(0.05)
                except Exception:
                    time.sleep(0.05)

    def stop(self):
        self.lock.acquire()
```

```
        self.continueLogging = False
        self.lock.release()

    # run function - required for threads, will run the logging thread using
    # record data.  Quits when self.continueLogging is false.
    def run(self):
        self.recordData()
        self.sock.close()
```

Data Logging Rate

The 510 FasTac chassis will log data at a rate of 0.25 Hz (once every 4 seconds) by default. To log data at a faster rate, the following process can be used:
(Line numbers and paths could vary depending on the version of software on the platform.)

First, change the speed at which the ChassisMonitor thread runs

On line 348 in file
/opt/irobot/bin/python/site-package/PackBotChassis/scripts/chassisDevices.py you will see some code like this:

```
        self.addThread(
                aware.PeriodicThread.narrow(
                        self.chassisMonitor.get("PollThread")))
```

Add a line that looks like this right under it

```
aware.PeriodicThread.narrow(self.chassisMonitor.get("PollThread")).setPeriod(aware.Duration(1
.0))
```

Change the duration of the thread period, in seconds, to your desired value. For this testing, a value of 0.25 was used. Running it too fast could have negative impacts on the rest of the system.

Now, change the data thread speed. Lines 151 - 154 of
opt/irobot/lib/python2.5/site-packages/PackBotHardwareProfile/combinedMotion/chassisEodMotion.py

```
        self.pubthread.connect('Action', self.pubgrp.get('ActionIF'))

        pt =
aware.PeriodicThread.narrow(self.pubthread.get('PeriodicThread'))
        pt.setPeriod(aware.Duration(4.0))
        pt.setPolicy(aware.PeriodicThreadPolicy.AS_AVAILABLE)

        self.addThread(pt)
```

Change line 153 to a duration smaller than 4.0, in seconds. For this testing, a value of 0.25 was used.

For FasTac, the path is the same but the file is chassisFasTacMotion.py, and the line number is 159.

APPENDIX D. SAMPLE TEST SHEET

Test Name			
Date		**LeftFront Batt S/N**	
Time of Start		**LeftRear Batt S/N**	
Time End			
Test Run #			
Robot ID		**Aware 2.0 Version**	
OCU S/N		**Speed**	
Standoff			
Test Plan	Small_robot_mission_profiles_V09		

General Observations

	File Name		**Comments**
Pea Gravel			
Sand			
Crushed Concrete			
Hill			
Obstacle Course			
Manipulator			

APPENDIX E. CSV DATA FILE PROCESSING MACRO

This is Version 06 of the CSV Battery Testing MS Excel macro, written in Visual Basic.

```
Attribute VB_Name = "Module1"
Sub ProcessBatt_CSVFile()
Attribute ProcessBatt_CSVFile.VB_Description = "Processes csv files captured during battery field testing.  \nWritten: 05-17-2010\nUpdated: 05-17-2010\nTy Valascho"
Attribute ProcessBatt_CSVFile.VB_ProcData.VB_Invoke_Func = "p\n14"
' Version 06
'
' ProcessBatt_CSVFile Macro
' Processes csv files captured during battery field testing.
' Written: 05-17-2010
' Updated: 07-22-2010
' Ty Valascho
'
'


    Dim MANIPULATOR_TEST
    ' The manipulator test does not have any movement (of the platform)
    ' so the calculations are different that the other tests.
    MANIPULATOR_TEST = False

    Dim Msg, Style, Title, Help, Ctxt, Response, MyString
    Msg = "Is this file from a manipulator test?" + Chr(13) + "**The manipulator test does not move the platform," + Chr(13) + "and the calculations are performed differently than the other tests.**" ' Define message.
    Style = vbYesNo    ' Define buttons.
    Title = "'Test Type"    ' Define title.
    Response = MsgBox(Msg, Style, Title)
    If Response = vbYes Then    ' User chose Yes.
       MANIPULATOR_TEST = True
    Else    ' User chose No.
       MANIPULATOR_TEST = False
    End If


    Application.ScreenUpdating = False
    Rows("1:4").Select
    Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    Columns("B:B").Select
    Selection.Insert Shift:=xlToRight, CopyOrigin:=xlFormatFromLeftOrAbove

    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Total Time (seconds)"
    Range("B2").Select
    ActiveCell.FormulaR1C1 = "Movement cal (m)"
    Range("C2").Select
    ActiveCell.FormulaR1C1 = 0.02

    Range("A4").Select
    ActiveCell.FormulaR1C1 = "Time (seconds)"
    Range("B4").Select
    ActiveCell.FormulaR1C1 = "Elapsed Time (sec)"
    Range("C4").Select
    ActiveCell.FormulaR1C1 = "LF Voltage (v)"
    Range("D4").Select
    ActiveCell.FormulaR1C1 = "LF Energy (j)"
    Range("E4").Select
    ActiveCell.FormulaR1C1 = "LF Power (w)"
    Range("F4").Select
    ActiveCell.FormulaR1C1 = "LF Temp (C)"
    Range("G4").Select
    ActiveCell.FormulaR1C1 = "LR Voltage (v)"
    Range("H4").Select
    ActiveCell.FormulaR1C1 = "LR Energy (j)"
    Range("I4").Select
    ActiveCell.FormulaR1C1 = "LR Power (w)"
    Range("J4").Select
    ActiveCell.FormulaR1C1 = "LR Temp (C)"
    Range("K4").Select
    ActiveCell.FormulaR1C1 = "Heading"
    Range("L4").Select
    ActiveCell.FormulaR1C1 = "Pitch"
```

```
    Range("M4").Select
    ActiveCell.FormulaR1C1 = "Roll"
    Range("N4").Select
    ActiveCell.FormulaR1C1 = "Translate Command"
    Range("O4").Select
    ActiveCell.FormulaR1C1 = "Xposition"
    Range("P4").Select
    ActiveCell.FormulaR1C1 = "Yposition"
    Range("Q4").Select
    ActiveCell.FormulaR1C1 = "Rotate"
    Range("R4").Select
    ActiveCell.FormulaR1C1 = "CAM Mtr Volts (v)"
    Range("S4").Select
    ActiveCell.FormulaR1C1 = "Flipper Mtr Volts (v)"
    Range("T4").Select
    ActiveCell.FormulaR1C1 = "SAM Mtr Volts (v)"
    Range("U4").Select
    ActiveCell.FormulaR1C1 = "Track Mtr Volts (v)"
    Range("V4").Select
    ActiveCell.FormulaR1C1 = "Left Track Curr (A)"
    Range("W4").Select
    ActiveCell.FormulaR1C1 = "Right Track Curr (A)"
    Range("X4").Select
    ActiveCell.FormulaR1C1 = "LF Curr Calculated (A) [P/v]"
    Range("Y4").Select
    ActiveCell.FormulaR1C1 = "LR Curr Calculated (A) [P/v]"
    Range("Z4").Select
    ActiveCell.FormulaR1C1 = "LF+LR Curr (A)"
    Range("AA4").Select
    ActiveCell.FormulaR1C1 = "LTrack+Rtrack Curr (A)"
    Range("AB4").Select
    ActiveCell.FormulaR1C1 = "System Voltage (V)"
    Range("AC4").Select
    ActiveCell.FormulaR1C1 = "Movement Bit"
    Range("AD4").Select

    If (MANIPULATOR_TEST = False) Then
       ActiveCell.FormulaR1C1 = "Current used for movement (A)"
       Range("AE4").Select
       ActiveCell.FormulaR1C1 = "Steady State Bit (V)"
       Range("AF4").Select
       ActiveCell.FormulaR1C1 = "Non-Movement Current (A)"
       Range("AF1").Select
       ActiveCell.FormulaR1C1 = "Avg steady-state current (A)"
    End If 'End MANIPULATOR_TEST = False

    Range("AD1").Select
    ActiveCell.FormulaR1C1 = "Sum (int)"
    Range("AE1").Select
    ActiveCell.FormulaR1C1 = "Avg motion current (A)"

    Range("B5").Select
    ActiveCell.FormulaR1C1 = 0

    Dim Check, Counter
'   Time (seconds)
'   Copy formulas to column B, starting from the bottom and going up
    Range("A4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select

    ActiveCell.FormulaR1C1 = "=RC[-1]-R5C[-1]"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
       Counter = Counter + 1    ' Increment Counter.
       If Counter > 10000 Then    ' If condition is True.
          Check = False    ' Set value of flag to False.
       End If
       ActiveCell.Offset(-1, 0).Select
       If ActiveCell.Value = "" Then
          ActiveCell.FormulaR1C1 = "=RC[-1]-R5C[-1]"
       Else
          Check = False
       End If
```

```
    Loop

'   LF Curr Calculated (A) [P/v]
'   Copy formulas to column X, starting from the bottom and going up
    Range("W4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
    ActiveCell.FormulaR1C1 = "=IF(OR(RC[-19]=""Invalid"",RC[-21]=""Invalid""),0,RC[-19]/RC[-21])"
   ' ActiveCell.FormulaR1C1 = "=RC[-19]/RC[-21]"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
       Counter = Counter + 1    ' Increment Counter.
       If Counter > 10000 Then    ' If condition is True.
          Check = False    ' Set value of flag to False.
       End If
       ActiveCell.Offset(-1, 0).Select
       If ActiveCell.Value = "" Then
          ActiveCell.FormulaR1C1 = "=IF(OR(RC[-19]=""Invalid"",RC[-21]=""Invalid""),0,RC[-19]/RC[-21])"
       Else
          Check = False
       End If
    Loop

'   LR Curr Calculated (A) [P/v]
'   Copy formulas to column Y, starting from the bottom and going up
    Range("X4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
'    ActiveCell.FormulaR1C1 = "=RC[-16]/RC[-18]"
    ActiveCell.FormulaR1C1 = "=IF(OR(RC[-16]=""Invalid"",RC[-18]=""Invalid""),0,RC[-16]/RC[-18])"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
       Counter = Counter + 1    ' Increment Counter.
       If Counter > 10000 Then    ' If condition is True.
          Check = False    ' Set value of flag to False.
       End If
       ActiveCell.Offset(-1, 0).Select
       If ActiveCell.Value = "" Then
          ActiveCell.FormulaR1C1 = "=IF(OR(RC[-16]=""Invalid"",RC[-18]=""Invalid""),0,RC[-16]/RC[-18])"
       Else
          Check = False
       End If
    Loop

'   LF+LR Curr (A)
'   Copy formulas to column Z, starting from the bottom and going up
    Range("Y4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
    ActiveCell.FormulaR1C1 = "=RC[-1]+RC[-2]"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
       Counter = Counter + 1    ' Increment Counter.
       If Counter > 10000 Then    ' If condition is True.
          Check = False    ' Set value of flag to False.
       End If
       ActiveCell.Offset(-1, 0).Select
       If ActiveCell.Value = "" Then
          ActiveCell.FormulaR1C1 = "=RC[-1]+RC[-2]"
       Else
          Check = False
       End If
    Loop

'   LTrack+Rtrack Curr (A)
'   Copy formulas to column AA, starting from the bottom and going up
    Range("Z4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select

    ActiveCell.FormulaR1C1 = "=ABS(RC[-5])+ABS(RC[-4])"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
       Counter = Counter + 1    ' Increment Counter.
```

```
    If Counter > 10000 Then    ' If condition is True.
      Check = False    ' Set value of flag to False.
    End If
    ActiveCell.Offset(-1, 0).Select
    If ActiveCell.Value = "" Then
      ActiveCell.FormulaR1C1 = "=ABS(RC[-5])+ABS(RC[-4])"
    Else
      Check = False
    End If
  Loop

'  System Voltage (V)
'  Copy formulas to column AB, starting from the bottom and going up
  Range("AA4").End(xlDown).Select
  ActiveCell.Offset(0, 1).Select

  ActiveCell.FormulaR1C1 = "=IF(OR(RC[-21]=""Invalid"",RC[-21]=26),RC[-25],IF(OR(RC[-25]=""Invalid"",RC[-25]=26),RC[-21],MAX(RC[-25],RC[-21])))"
  Check = True
  Counter = 0    ' Initialize variables.
  Do While Check = True
    Counter = Counter + 1    ' Increment Counter.
    If Counter > 10000 Then    ' If condition is True.
      Check = False    ' Set value of flag to False.
    End If
    ActiveCell.Offset(-1, 0).Select
    If ActiveCell.Value = "" Then
      ActiveCell.FormulaR1C1 = "=IF(OR(RC[-21]=""Invalid"",RC[-21]=26),RC[-25],IF(OR(RC[-25]=""Invalid"",RC[-25]=26),RC[-21],MAX(RC[-25],RC[-21])))"
    Else
      Check = False
    End If
  Loop

'  Movement Bit: Goes TRUE when the platform is moving
'  Copy formulas to column AC, starting from the bottom and going up
  Range("AB4").End(xlDown).Select
  ActiveCell.Offset(0, 1).Select
  ActiveCell.FormulaR1C1 = "=IF(AND(ABS(RC[-13]-R[-1]C[-13])<R2C3,ABS(RC[-14]-R[-1]C[-14])<R2C3),0,1)"
  Check = True
  Counter = 0    ' Initialize variables.
  Do While Check = True
    Counter = Counter + 1    ' Increment Counter.
    If Counter > 10000 Then    ' If condition is True.
      Check = False    ' Set value of flag to False.
    End If
    ActiveCell.Offset(-1, 0).Select
    If ActiveCell.Value = "" Then
      ActiveCell.FormulaR1C1 = "=IF(AND(ABS(RC[-13]-R[-1]C[-13])<R2C3,ABS(RC[-14]-R[-1]C[-14])<R2C3),0,1)"
    Else
      ActiveCell.Offset(1, 0).Select
      ActiveCell.Value = 0
      Check = False
    End If
  Loop

  If (MANIPULATOR_TEST = False) Then

  '  Current used for movement: the amount of current used for movement only
  '  (when the movement bit is true)
  '  Copy formulas to column AD, starting from the bottom and going up
    Range("AC4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*RC[-3]"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
      Counter = Counter + 1    ' Increment Counter.
      If Counter > 10000 Then    ' If condition is True.
        Check = False    ' Set value of flag to False.
      End If
      ActiveCell.Offset(-1, 0).Select
      If ActiveCell.Value = "" Then
        ActiveCell.FormulaR1C1 = "=RC[-1]*RC[-3]"
      Else
        ActiveCell.Offset(1, 0).Select
```

```
        ActiveCell.Value = 0
        Check = False
      End If
    Loop

'   Steady State Bit: Indicates the platform is not moving and the current is less than 1.7 amps
'   - this value is when the platform is just sitting there, doing nothing.
'   Copy formulas to column AE, starting from the bottom and going up
    Range("AD4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
    ActiveCell.FormulaR1C1 = "=IF(AND(NOT(RC[-2]),RC[-4]<1.7),1,0)" '=IF(AND(NOT(AC5),Z5<1.7),1,0)
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
      Counter = Counter + 1    ' Increment Counter.
      If Counter > 10000 Then    ' If condition is True.
        Check = False    ' Set value of flag to False.
      End If
      ActiveCell.Offset(-1, 0).Select
      If ActiveCell.Value = "" Then
        ActiveCell.FormulaR1C1 = "=IF(AND(NOT(RC[-2]),RC[-4]<1.7),1,0)"
      Else
        ActiveCell.Offset(1, 0).Select
        ActiveCell.Value = 0
        Check = False
      End If
    Loop

'   Non-Movement Current: Amount of current used when platform is idle.
'   Copy formulas to column AF, starting from the bottom and going up
    Range("AE4").End(xlDown).Select
    ActiveCell.Offset(0, 1).Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*RC[-5]"
    Check = True
    Counter = 0    ' Initialize variables.
    Do While Check = True
      Counter = Counter + 1    ' Increment Counter.
      If Counter > 10000 Then    ' If condition is True.
        Check = False    ' Set value of flag to False.
      End If
      ActiveCell.Offset(-1, 0).Select
      If ActiveCell.Value = "" Then
        ActiveCell.FormulaR1C1 = "=RC[-1]*RC[-5]"
      Else
        ActiveCell.Offset(1, 0).Select
        ActiveCell.Value = 0
        Check = False
      End If
    Loop

'Create Averages and summary values
    Range("AD5").End(xlDown).Select
    ActiveCell.Name = "End_Cell_Movement_Current"
    Range("AF5").End(xlDown).Select
    ActiveCell.Name = "End_Cell_Non_Movement_Current"

    Range("AD2").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[3]C:End_Cell_Movement_Current)"
    Range("AE2").Select
    ActiveCell.FormulaR1C1 = "=AVERAGEIF(R[3]C[-1]:End_Cell_Movement_Current,""<>0"")"
    Range("AF2").Select
    ActiveCell.FormulaR1C1 = "=AVERAGEIF(R[3]C:End_Cell_Non_Movement_Current,""<>0"")"

    Range("AD2:AF2").Select
    Selection.Copy
    Range("AD3").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
      :=False, Transpose:=False

   ' Add total time of test
    Range("AC5").End(xlDown).Select
    ActiveCell.Name = "End_Cell_Movement_Bit"
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "=(SUM(R[4]C[26]:End_Cell_Movement_Bit))*.250"
```

```
    'End MANIPULATOR_TEST = False

    Else 'MANIPULATOR_TEST = True
    'Create Averages and summary values
       Range("Z5").End(xlDown).Select
       ActiveCell.Name = "End_Cell_LF_LR_Curr"

       Range("AD2").Select
       ActiveCell.FormulaR1C1 = "=SUM(R[3]C[-4]:End_Cell_LF_LR_Curr)"
       Range("AE2").Select
       ActiveCell.FormulaR1C1 = "=AVERAGE(R[3]C[-5]:End_Cell_LF_LR_Curr)"

       Range("AD2:AE2").Select
       Selection.Copy
       Range("AD3").Select
       Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
          :=False, Transpose:=False

      ' Add total time of test
       Range("B5").End(xlDown).Select
       Selection.Copy
       Range("C1").Select
       Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
          :=False, Transpose:=False


    End If 'End MANIPULATOR_TEST = True

' Clean up, make pretty
    Range("C5").Select
    ActiveWindow.FreezePanes = True
    Rows("4:4").Select
    Selection.Font.Bold = True
    Range("B1").Select
    Selection.Font.Bold = True
    With Selection
       .HorizontalAlignment = xlRight
       .VerticalAlignment = xlBottom
       .WrapText = False
       .Orientation = 0
       .AddIndent = False
       .IndentLevel = 0
       .ShrinkToFit = False
       .ReadingOrder = xlContext
       .MergeCells = False
    End With


'Create Graphs
' We don't know where the end of the columns are, so we must find the end
' and name it so we can select that range for the graphs

    Range("B4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_Elapsed_Time"

' Create Energy Consumption graph
' Version 06 removed this - energy consumption is so inaccurate,
' this graph is meaningless
'    Range("D4").End(xlDown).Select
'    ActiveCell.Name = "End_Cell_LF_Energy"
'    Range("H4").End(xlDown).Select
'    ActiveCell.Name = "End_Cell_LR_Energy"
'    ActiveSheet.Shapes.AddChart.Select
'    ActiveChart.ChartType = xlXYScatterLinesNoMarkers
'    ActiveChart.SetSourceData Source:=Range("H4:End_Cell_LR_Energy, B4:End_Cell_Elapsed_Time")
'    ActiveChart.SeriesCollection.Add Source:=Range("D4:End_Cell_LF_Energy"), Rowcol:=2, SeriesLabels:=True
'    ActiveChart.Location Where:=xlLocationAsNewSheet, Name:= _
'       "Energy Consumption"

' Create Current Consumption graph
'    Sheets(1).Activate ' Go back to the spreadsheet
    Range("X4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_LF_Curr_Calculated"
```

```
    Range("Y4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_LR_Curr_Calculated"
    Range("Z4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_LF_LR_Curr"
    Range("AA4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_LTrack_Rtrack_Curr"
    Range("AC4").End(xlDown).Select
    ActiveCell.Name = "End_Cell_Movement_Bit"

    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.ChartType = xlXYScatterLinesNoMarkers
    ActiveChart.SetSourceData Source:=Range("AC4:End_Cell_Movement_Bit, B4:End_Cell_Elapsed_Time") '"), Rowcol:=2, SeriesLabels:=True
    ActiveChart.SeriesCollection.Add Source:=Range("Z4:End_Cell_LF_LR_Curr"), Rowcol:=2, SeriesLabels:=True
    ActiveChart.SeriesCollection.Add Source:=Range("AA4:End_Cell_LTrack_Rtrack_Curr"), Rowcol:=2, SeriesLabels:=True

    ActiveChart.Location Where:=xlLocationAsNewSheet, Name:= _
       "Current Consumption"

    Application.ScreenUpdating = True



    Pathname = ActiveWorkbook.Path
    Filename = ActiveWorkbook.Name
    ' Remove file extension
    Filename = Left(Filename, (Len(Filename) - 4))
    Filename = Filename + " processed.xlsm"
    ActiveWorkbook.SaveAs Filename:= _
       (Pathname + "\" + Filename) _
       , FileFormat:=xlOpenXMLWorkbookMacroEnabled, CreateBackup:=False
'    Application.Goto Reference:="PERSONAL.XLSB!save_file"

End Sub
```